Head pose Estimation Using Convolutional Neural Networks

Xingyu Liu
June 6, 2016
xyl@stanford.edu

Abstract

Head pose estimation is a fundamental problem in computer vision. Several methods has been proposed to solve this problem. Most existing methods use traditional computer vision methods and existing method of using neural networks works on depth bitmaps.

In this project, we explore using convolutional neural networks (CNNs) that take RGB image as input to estimate the head pose. We use regression as the estimation approach. We explored the effect of different regularization strength and face alignment in our estimation. By using a CNN whose architecture is similar to VGG-nagadomi to train on IHDB head pose dataset, we can get a test regression euclidean loss of less than 0.0113, equivalent to average error of 20° of spherical distance, 4 times smaller than not using face alignment. We also proved that proper regularization strength could prevent overfitting thus reduce test loss.

1. Introduction

The estimation for head pose has been a fundamental and promising problem in computer vision. It is an important problem of understanding 3D scene from the 2D image. It can also be used in various situations that needs analysis of human behavior. Precise estimation of head pose could result in revolutionary change in human-computer interaction applications and virtual reality. For example, there has already been mobile games that uses head pose information as a way of control [12]. We expect the industry of HCI could utilize the result from precisely estimating the head pose as a way to provide better service. Since extremely deep convolutional neural network has been pushing the recognition task to way better results than people originally imagine, we expect using deep convolutional neural network could also result in better head pose estimation result and could potentially have great impact on the computer vision, HCI industry and other related fields.

2. Related Works

2.1. Previous Works

Several previous research works on estimating head pose has been published. Breitenstein *et al.* [2] introduced an algorithm for real-time face pose estimation of the 3D pose of a previously unseen face from a single range image. They uses novel shape signature to identify noses in range images. Their method include generating and evaluating many pose hypotheses using GPU. They also developed a novel error function that compares the input range image to precomputed pose images of an average face model. Their algorithm was evaluated on a database of range images with large pose variations developed by a method for automatic ground truth annotation.

Brown *et al.* [3] proposed a method of image-based learning, estimation of a wide range of pose, and is capable of real-time performance for lowresolution imagery.

Murphy *et al.* [8] summarized the existing approaches and works in head pose estimation before 2009. The works discussed are all traditional applied machine learning methods.

Fanelli *et al.* [4] used an approach based on discriminative random regression forests: ensembles of random trees trained by splitting each node so as to simultaneously reduce the entropy of the class labels distribution and the variance of the head position and orientation. They evaluated three different approaches to jointly take classification and regression performance into account during training.

Li *et al.* [7] propose a deep convolutional neural network for 3D human pose estimation from monocular images. Their neural network is trained using two strategies:

1) a multi-task framework that jointly trains pose regression and body part detectors; 2) a pre-training strategy where the pose regressor is initialized using a network trained for body part detection.

2.2. Our Approach

As we can see, previous work on head pose estimation has been using traditional computer vision and applied machine learning approaches. Recent years, convolutional neural network (CNN) has been successful in recognition and classification tasks. Applying CNN in head pose estimation may beat traditional algorithms.

A recent work [13] uses CNN that takes depth information collected from depth sensors as input to estimate the head pose. However, using depth information might limit the application scenario, since high-quality depth sensors is still not ubiquitous.

We propose to use RGB images collected from cameras directly as the source of input. It could be a much more economic approach compared to depth sensor. Moreover, working on RGB images itself is a more interesting problem of computer vision.

3. Methodologies

3.1. Overview

We used IHDB Head Pose dataset [10] to train and test my CNN. The CNN has an architecture similar to VGG-nagadomi [1]. We tried both training with and without face alignment and obtained both results. Different training hyperparameters such as regularization strength and learning rate etc are also explored.

3.2. Details

3.2.1 Training and Testing Dataset Preprocessing

The IDIAP Head Pose dataset [10] consists of 21,152 75×75 images from 18 people. The faces are flipped left-right together with the labels so that the effective number of face is doubled. The faces are pretty much located near the center of the image. However, the face detection and alignment may still be needed.

Since some of the faces are in profile position to the camera, we used both the Haar Cascade Classifier for front face and the Haar Cascade Classifier for profile face to first detect the face. For most images in the dataset, at least and at most one face will be detected and a square will be given specifying the position of the face. The Haar Cascade Classifier cannot detect faces in a small portion of images due to occlusion. Those images are discarded. After detection of face-containing square subimages, the subimages are resized to a fixed constant size to feed into the CNN.

The head pose is described in three angles: pan, tilt and roll, where pan and tilt discribe the direction of looking and roll discribes the rotation around the neck. This result in each image in the dataset contains three labels. The three labels are normalized to the value within $[-1, 1]$. We plotted the distribution of the three angles in 3D space in figure 1. As we expected, the labels are symetric with respect to pan axis. That's because the training dataset is augmented by flip left and right. The example of the training image is shown in figure 2.

Some of the images in the training and testing dataset is occluded, so we should treat them as irrelevant data point. Figure 3 shows the example of occlusion. The images that doesn't contain faces or the Haar Cascade Classifier cannot detect faces from will be deleted from training and testing set. Figure 4 shows examples of successfully detected by Haar Cascase Classifier and the aligned faces in the examples in Figure 2

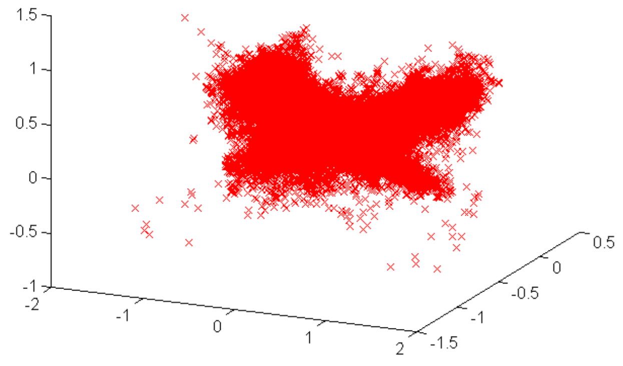

Figure 1. Angle distribution

3.2.2 CNN Architecture

We plan to use a similar CNN architecture as VGG-11 Nagadomi [1]. VGG-11 Nagadomi contains three groups of convolution layers, each consisting of three convolution layers with the same kernel size of 3 but different channel depth among groups. The convolution layers are followed by sev-

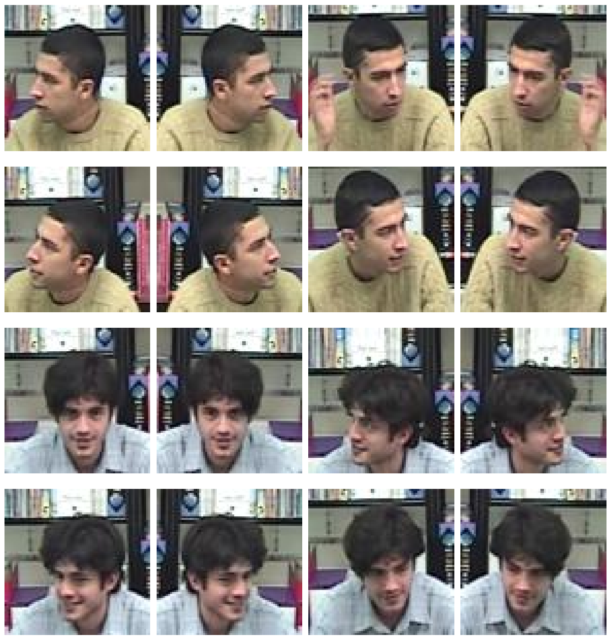
Figure 2. Example training data

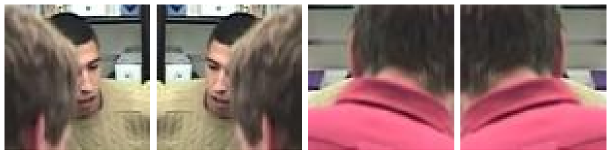
Figure 3. Bad Example of face detection because of occlusion training data

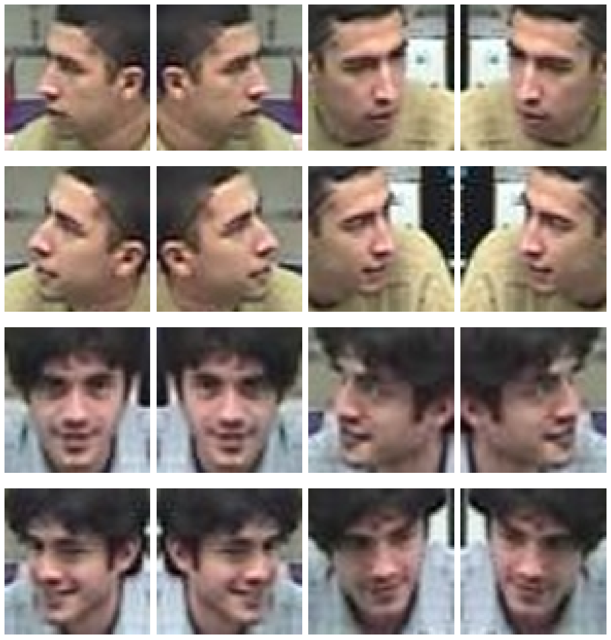
Figure 4. Good Example of face detection and face alignment

eral fully-connected layers. The reason we chose VGG-11 Nagadomi is that, unlike ImageNet networks, VGG-11 Nagadomi works on CIFAR dataset [6], which means it's large enough for small classification problems and should be enough for head pose estimation.

One fundamental problem is whether to use classification or regression as the estimation method. In the first approach, i.e. classification, the output score of the CNN is the probability of rotation angles in each bin. In the second approach, i.e. regression, the output score of the CNN is the three angles. The difference is that, in classification, the relation between bins are all independent and the same, while actually it shouldn't. The angle values that are closer should have closer relations. Distinguishing the relation of different bin pairs can also improve learning, since the models that give bad prediction can learn to adjust itself faster because of larger gradient. So we chose to use regression, where the loss could be obtained from the Euclidean distance between the predicted scores and the groundtruth labels. The final CNN architecture we use is as follows:

$$2\times(64Conv3+ReLU) \rightarrow MP2 \rightarrow 2\times(128Conv3+ReLU)$$

$$\rightarrow MP2 \rightarrow 4\times(256Conv3+ReLU) \rightarrow MP2$$

$$\rightarrow 2\times(FC1024+ReLU) \rightarrow FC3$$

3.2.3 CNN training

Since we're using regression instead of classification, the metrics we used to measure how good my model is no longer the training loss and testing accuracy as in the classification case. The Euclidean distance can be used to measure how close the prediction to the groundtruth and at the same time, it's also used in training as the loss function. So we used the Euclidean distance as loss in the training and testing measurement in the testing.

Besides the pre-processing of training and testing images, the selection of training hyperparameter of CNN is crucial for getting good performace on estimation accuracy. In this project, we used the default VGG11 Nagadomi training parameters like learning rate, learning type (Step) and tuned the hyperparameter of regularization strength. we used the regularization strength in the equal logarithmic interval and see how good the testing loss each model can get a test regression euclidean loss of less than 0.0113, equivalent to average error of 20° of spherical distance. This is conservative estimation, since the loss contains regularization terms and should be deducted.

4. Experiment

4.1. Experiment Setup

We used Caffe [5] as my CNN training framework. Since each of the images contains three labels instead of just one, We can only use HDF5 data type for training input instead of the default IMDB type. We used a base learning rate of 0.0005, learning type of ”step”, a step size of $step\_size = 10,000$ and a step factor of $\gamma = 0.7$. The step factor means the factor that times the learning rate after $step\_size$ iterations of training. For weight update, we use Nesterov’s accelerated gradient [9]. As we have discussed earlier, the training and testing loss itself is the measurement of how good our model is.

For image preprocessing, we prepared trainng and testing data and implemented the face detection using the python interface of OpenCV v2 and **Haar Cascade Classifier** [11].

4.2. Experiment Result

When using unaligned face with regularization strength of 0.0005, the training and testing loss are shown in Figure 5. When using unaligned face with regularization strength of 0.005, the training and testing loss are shown in Figure 6. When using aligned face with regularization strength of 0.005, the training and testing loss are shown in Figure 7. When using aligned face with regularization strength of 0.0005, the training and testing loss are shown in Figure 8. When using aligned face with no regularization strength, the training and testing loss are shown in Figure 9. As we can see, when using aligned face to train and test and using a regularization of 0.0005, we can get a regularization

From the above results, we can see that aligning of face faces plays extremely important role in improving test performance of head pose estimation. Comparing Figure 5 and 6 with Figure 7, 8 and 9, we can find that the testing loss can be reduced to a quarter of the original testing loss if the face is aligned. The reason behind this result is pretty straightforward. If we didn’t align the images to the faces, we are taking the background into account and that will distract the CNN from learning the essential information which is the face area in the image.

We can also compare the learning curve of the testing loss to the learning curve of the training loss in Figure 7, 8 and 9. As we can see, with proper regularization strength, the testing loss is closer to the the training loss, which means there’re less overfitting and higher testing performance. Comparing the absolute value of loss is meaningless, since with different regularization values, the loss has different regularization loss terms.

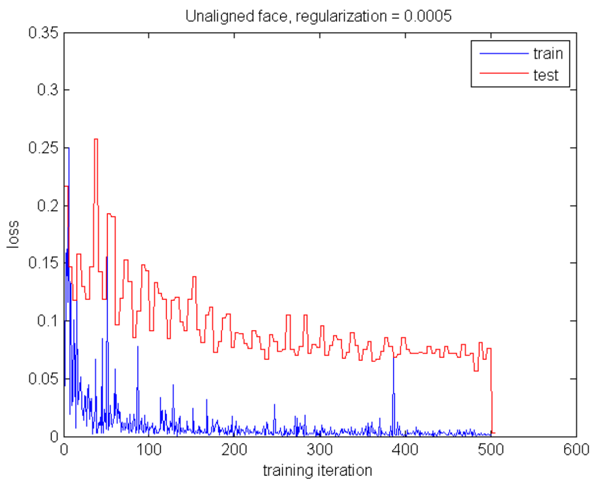

Figure 5. Unaligned face, with regularization strength of 0.0005

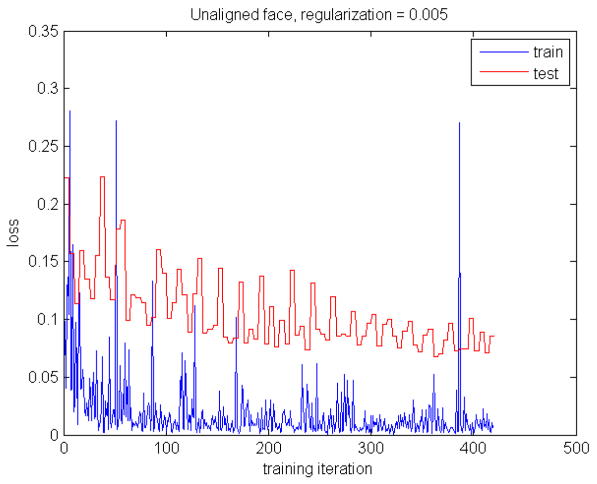

Figure 6. Unaligned face, with regularization strength of 0.005

Here we show some results of pose estimatin. The good prediction results are illustrated in Figure 10. The bad prediction results are illustrated in Figure **??**. It shows that good prediction can be more easily obtained from heads facing near front direction. The extreme profile faces and down faces are most easily get bad result. The reason is also straightforward: the profile faces and down faces contain less facial feature and information comparing to the front faces. Thus the CNN is more inclined to give erroneous results.

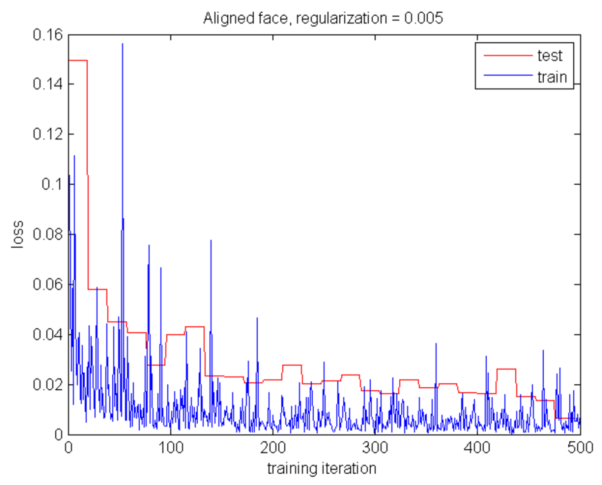

Figure 7. Aligned face, with regularization strength of 0.005

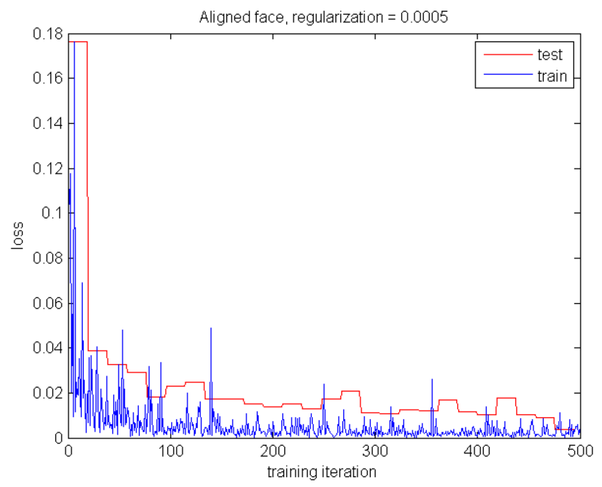

Figure 8. Aligned face, with regularization strength of 0.0005

5. Conclusion

In this project, we explore using a convolutional neural networks similar to VGG-11 nagadomi to estimate the head pose. We use regression as the estimation approach. We explored the effect of different regularization strength and face alignment in our estimation. When using face alignment, we can get a test regression euclidean loss of less than 0.0113, equivalent to average error of 20° of spherical distance, 4 times smaller than not using face alignment. We also proved that proper regularization strength could prevent overfitting thus reduce test loss.

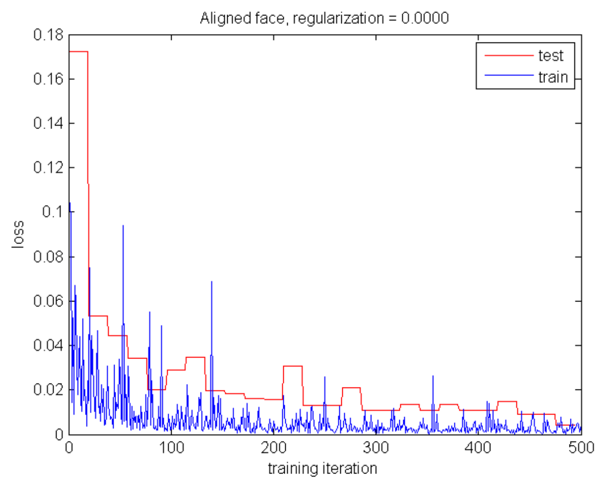

Figure 9. Aligned face, with regularization strength of 0.0000

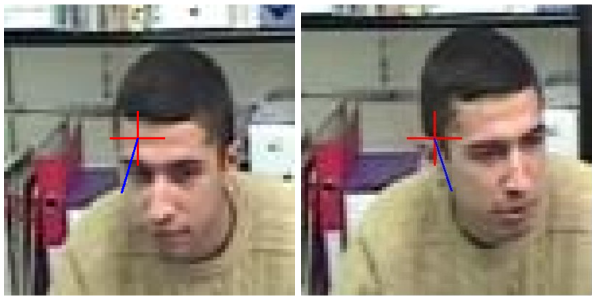

Figure 10. Good Result of Head pose estimation

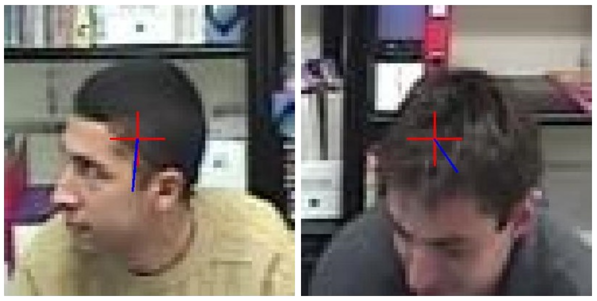

Figure 11. Bad Result of Head pose estimation

References

[1] Code for kaggle-cifar10 competition. 5th place. `https://github.com/nagadomi/kaggle-cifar10-torch7`.

[2] Michael D Breitenstein, Daniel Kuettel, Thibaut Weise, Luc Van Gool, and Hanspeter Pfister. Real-time face pose estimation from single range images. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[3] Lisa M Brown and Ying-Li Tian. Comparative study of coarse head pose estimation. In *Motion and Video Computing, 2002. Proceedings. Workshop on*, pages 125–130. IEEE, 2002.

[4] Gabriele Fanelli, Thibaut Weise, Juergen Gall, and Luc Van Gool. Real time head pose estimation from consumer depth cameras. In *Pattern Recognition*, pages 101–110. Springer, 2011.

[5] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[6] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009.

[7] Sijin Li and Antoni B Chan. 3d human pose estimation from monocular images with deep convolutional neural network. In *Computer Vision–ACCV 2014*, pages 332–347. Springer, 2014.

[8] Erik Murphy-Chutorian and Mohan Manubhai Trivedi. Head pose estimation in computer vision: A survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(4):607–626, 2009.

[9] Yurii Nesterov et al. Gradient methods for minimizing composite objective function. Technical report, UCL, 2007.

[10] Diego Tosato, Mauro Spera, Matteo Cristani, and Vittorio Murino. Characterizing humans on riemannian manifolds. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1972–1984, 2013.

[11] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[12] Wenbin Tang et al. Crows Coming. `https://itunes.apple.com/us/app/crows-coming/id452523144?mt=8`.

[13] Zhiang Hu. Real-Time Head Pose Estimation with Convolutional Neural Networks. `http://cs231n.stanford.edu/reports/zhianghu_report.pdf`.